

What is claimed is:

- 1 1. A computerized method comprising, in each of a plurality of processors  
2 including a first processor and a second processor:
  - 3 (a) loading a first vector register with addressing values;
  - 4 (b) loading a second vector register with operand values;
  - 5 (c) determining which, if any, element addresses of the first vector  
6 register have a value that duplicates a value in another element address; and
  - 7 (d) selectively adding certain elements of the second vector of operand  
8 values based on the element addresses the duplicated values;
  - 9 (e) loading, using indirect addressing from the first vector register,  
10 elements from memory into a third vector register;
  - 11 (f) adding values from the third vector register and the second vector of  
12 operand values to generate a result vector; and
  - 13 (g) storing the result vector to memory using indirect addressing.

- 1 2. The method of claim 1, wherein the set of operations (a), (b), (c), and (d) is  
2 performed substantially in parallel in the plurality of processors, and the set of  
3 operations (e), (f), and (g) is performed serially, one processor at a time.

- 1 3. The method of claim 1, further comprising  
2 executing an ordered Msync operation before the set of operations (e), (f),  
3 and (g); and  
4 executing an end ordered Msync operation after the set of operations (e), (f),  
5 and (g).

- 1 4. The method of claim 3, wherein the set of operations (a), (b), (c), and (d) is  
2 performed substantially in parallel in the plurality of processors.

- 1 5. The method of claim 1, further comprising  
2 executing a first barrier synchronization operation before the set of operations

3 (e), (f), and (g) in all of the plurality of processors;  
4       executing a second barrier synchronization operation before the set of  
5 operations (e), (f), and (g) in the second processor;  
6       executing the set of operations (e), (f), and (g) in the first processor and then  
7 executing a second barrier synchronization operation in the first processor to satisfy  
8 the second barrier synchronization in the second processor, and executing a third  
9 barrier synchronization in the first processor; and  
10       executing the set of operations (e), (f), and (g) in the second processor and  
11 then executing a third barrier synchronization operation in the second processor to  
12 satisfy the third barrier synchronization in the first processor.

1 6. The method of claim 5, wherein the set of operations (a), (b), (c), and (d) is  
2 performed substantially in parallel in the plurality of processors.

1 7. The method of claim 1,  
2 wherein the determining of duplicates includes:  
3       generating each respective address value for a sequence of addressed  
4 locations within a constrained area of memory containing  $2^N$  consecutive  
5 addresses using an N-bit value derived from each respective addressing  
6 value of the first vector register,  
7       generating each respective data value of a first sequence of values by  
8 combining at least a portion of each respective addressing value of the  
9 first vector register to a respective one of a sequence of integer numbers,  
10       storing the first sequence of values to the constrained memory area  
11 using the generated sequence of respective address values,  
12       loading a second first sequence of values from the constrained  
13 memory area using the generated sequence of respective address values,  
14 and  
15       comparing the first sequence of values to the second sequence of  
16 values; and  
17 wherein the loading of the third vector register includes loading elements

18 from locations specified by addressing values corresponding to indications of  
19 positive compares from the comparing;  
20 wherein addresses of the elements from memory are calculated by adding  
21 each respective addressing value to a base address;  
22 wherein the adding includes a floating-point addition operation that produces  
23 at least one element of the result vector as an ordered-operation floating point  
24 summation of an element of the loaded third vector register and a plurality of  
25 respective elements of the original second vector of operand values corresponding to  
26 elements of the first vector of addressing values having identical values, and  
27 wherein for the storing of the result vector of elements to memory, elements  
28 are stored to locations specified by addressing values corresponding to indications of  
29 positive compares.

1 8. A computerized method comprising:  
2 (a) within a first vector processor:  
3 loading a first vector register in the first vector processor with  
4 addressing values;  
5 loading a second vector register in the first vector processor with  
6 operand values;  
7 determining which, if any, element addresses of the first vector  
8 register in the first vector processor have a value that duplicates a value in  
9 another element address;  
10 selectively adding certain elements of the second vector of operand  
11 values in the first vector processor based on the element addresses the  
12 duplicated values;  
13 (b) within a second vector processor:  
14 loading a first vector register in the second vector processor with  
15 addressing values;  
16 loading a second vector register in the second vector processor with  
17 operand values;

18                   determining which, if any, element addresses of the first vector  
19                   register in the second vector processor have a value that duplicates a value  
20                   in another element address;  
21                   selectively operating on certain elements of the second vector of  
22                   operand values in the second vector processor based on the element  
23                   addresses the duplicated values;  
24       (c)       performing a synchronization operation that ensures that prior store  
25       operations effectively complete in at least the second vector processors before the  
26       following (d) operations;  
27       (d)       within the first vector processor:  
28                   loading, using indirect addressing from the first vector register,  
29                   elements from memory into a third vector register in the first vector  
30                   processor;  
31                   operating on values from the third vector register and the second  
32                   vector of operand values in the first vector processor to generate a first  
33                   result vector; and  
34                   storing the first result vector to memory using indirect addressing;  
35       (e)       performing a synchronization operation that ensures that the storing of the  
36       first result vector effectively completes before the following (f) operations; and  
37       (f)       within the second vector processor:  
38                   loading, using indirect addressing from the first vector register,  
39                   elements from memory into a third vector register in the second vector  
40                   processor;  
41                   operating on values from the third vector register and the second  
42                   vector of operand values in the second vector processor to generate a  
43                   second result vector; and  
44                   storing the second result vector to memory using indirect addressing.

1       9.       The method of claim 8, wherein each of the operating on functions includes  
2       adding.

1     10.     The method of claim 9, wherein the adding includes a floating-point addition  
2     operation that produces at least one element of the result vector as an ordered-  
3     operation floating point summation of an element of the loaded third vector register  
4     and a plurality of respective elements of the original second vector of operand values  
5     corresponding to elements of the first vector of addressing values having identical  
6     values.

1     11.     The method of claim 8,  
2     wherein the determining of duplicates includes:  
3             generating each respective address value for a sequence of addressed  
4             locations within a constrained area of memory containing  $2^N$  consecutive  
5             addresses using an N-bit value derived from each respective addressing  
6             value of the first vector register,  
7             generating each respective data value of a first sequence of values by  
8             combining at least a portion of each respective addressing value of the  
9             first vector register to a respective one of a sequence of integer numbers,  
10            storing the first sequence of values to the constrained memory area  
11            using the generated sequence of respective address values,  
12            loading a second first sequence of values from the constrained  
13            memory area using the generated sequence of respective address values,  
14            and  
15            comparing the first sequence of values to the second sequence of  
16            values.

1     12.     The method of claim 8, further wherein the loading of the third vector  
2     register of each processor includes loading elements from locations specified by  
3     addressing values corresponding to indications of positive compares from the  
4     comparing operation.

1     13.     The method of claim 8, further wherein indirect addresses of the elements  
2     from memory are calculated by adding each respective addressing value to a base

3 address.

1 14. A system comprising:

2 a first vector register having addressing values;

3 a second vector register having operand values;

4 circuitry programmed to determine which, if any, element addresses of the

5 first vector register have a value that duplicates a value in another element address;

6 circuitry programmed to selectively add certain elements of the second vector

7 of operand values based on the element addresses the duplicated values;

8 circuitry programmed to load, using indirect addressing from the first vector

9 register, elements from memory into a third vector register;

10 circuitry programmed to add values from the third vector register and the

11 second vector of operand values to generate a result vector; and

12 circuitry programmed to store the result vector to memory using indirect

13 addressing.

1 15. The system of claim 14,

2 wherein the circuitry programmed to determine duplicates includes:

3 circuitry programmed to generate each respective address value for a

4 sequence of addressed locations within a constrained area of memory

5 containing  $2^N$  consecutive addresses using an N-bit value derived from

6 each respective addressing value of the first vector register,

7 circuitry programmed to generate each respective data value of a first

8 sequence of values by combining at least a portion of each respective

9 addressing value of the first vector register to a respective one of a

10 sequence of integer numbers,

11 circuitry programmed to store the first sequence of values to the

12 constrained memory area using the generated sequence of respective

13 address values,

14 circuitry programmed to load a second sequence of values from the

15 constrained memory area using the generated sequence of respective

16 address values, and  
17 circuitry programmed to compare the first sequence of values to the  
18 second sequence of values; and  
19 wherein the circuitry programmed to load the third vector register loads  
20 elements from locations specified by addressing values corresponding to indications  
21 of positive compares;  
22 wherein addresses of the elements from memory are calculated by adding  
23 each respective addressing value to a base address;  
24 wherein the circuitry programmed to add includes a floating-point adder that  
25 produces at least one element of the result vector as an ordered-operation floating  
26 point summation of an element of the loaded third vector register and a plurality of  
27 respective elements of the original second vector of operand values corresponding to  
28 elements of the first vector of addressing values having identical values.

1 16. The system of claim 14, further comprising:  
2 circuitry programmed to perform the set of operations (a), (b), (c), and (d)  
3 substantially in parallel in the plurality of processors, and  
4 circuitry programmed to perform the set of operations (e), (f), and (g)  
5 serially, one processor at a time.

1 17. The system of claim 14, further comprising:  
2 circuitry programmed to execute an ordered Msync operation before the set  
3 of operations (e), (f), and (g); and  
4 circuitry programmed to execute an end ordered Msync operation after the set  
5 of operations (e), (f), and (g).

1 18. The system of claim 17, further comprising:  
2 circuitry programmed to perform the set of operations (a), (b), (c), and (d)  
3 substantially in parallel in the plurality of processors.

1 19. The system of claim 14, further comprising:

2           circuitry programmed to execute a first barrier synchronization operation  
3 before the set of operations (e), (f), and (g) in all of the plurality of processors;  
4           circuitry programmed to execute a second barrier synchronization operation  
5 before the set of operations (e), (f), and (g) in the second processor;  
6           circuitry programmed to execute the set of operations (e), (f), and (g) in the  
7 first processor and then executing a second barrier synchronization operation in the  
8 first processor to satisfy the second barrier synchronization in the second processor,  
9 and executing a third barrier synchronization in the first processor; and  
10          circuitry programmed to execute the set of operations (e), (f), and (g) in the  
11 second processor and then executing a third barrier synchronization operation in the  
12 second processor to satisfy the third barrier synchronization in the first processor.

1   20.    The system of claim 19, further comprising circuitry programmed to perform  
2 the set of operations (a), (b), (c), and (d) substantially in parallel in the plurality of  
3 processors.

1   21.    A system comprising:  
2   (a)    a first vector processor that includes:  
3           means for loading a first vector register in the first vector processor  
4           with addressing values;  
5           means for loading a second vector register in the first vector processor  
6           with operand values;  
7           means for determining which, if any, element addresses of the first  
8           vector register in the first vector processor have a value that duplicates a  
9           value in another element address;  
10          means for selectively adding certain elements of the second vector of  
11          operand values in the first vector processor based on the element addresses  
12          the duplicated values;  
13   (b)    a second vector processor that includes:  
14          means for loading a first vector register in the second vector processor  
15          with addressing values;



- 16 means for loading a second vector register in the second vector  
17 processor with operand values;  
18 means for determining which, if any, element addresses of the first  
19 vector register in the second vector processor have a value that duplicates  
20 a value in another element address;  
21 means for selectively operating on certain elements of the second  
22 vector of operand values in the second vector processor based on the  
23 element addresses the duplicated values;  
24 (c) means for performing a synchronization operation that ensures that prior  
25 store operations effectively complete in at least the second vector processors before  
26 the operations of the following (d) means;  
27 (d) within the first vector processor:  
28 means for loading, using indirect addressing from the first vector  
29 register, elements from memory into a third vector register in the first  
30 vector processor;  
31 means for operating on values from the third vector register and the  
32 second vector of operand values in the first vector processor to generate a  
33 first result vector; and  
34 means for storing the first result vector to memory using indirect  
35 addressing;  
36 (e) performing a synchronization operation that ensures that the storing of the  
37 first result vector effectively completes before the operations of the following (f)  
38 means; and  
39 (f) within the second vector processor:  
40 means for loading, using indirect addressing from the first vector  
41 register, elements from memory into a third vector register in the second  
42 vector processor;  
43 means for operating on values from the third vector register and the  
44 second vector of operand values in the second vector processor to generate  
45 a second result vector; and  
46 means for storing the second result vector to memory using indirect

47                   addressing.

1     22.     The system of claim 21, wherein each of the means for operating on  
2     functions includes an adder.

1     23.     The system of claim 22, further  
2             wherein the adder includes a floating-point adder that produces at least one  
3     element of the result vector as an ordered-operation floating point summation of an  
4     element of the loaded third vector register and a plurality of respective elements of  
5     the original second vector of operand values corresponding to elements of the first  
6     vector of addressing values having identical values

1     24.     The system of claim 23, wherein the means for determining of duplicates  
2     further includes:  
3             means for generating each respective address value for a sequence of  
4             addressed locations within a constrained area of memory containing  $2^N$   
5             consecutive addresses using an N-bit value derived from each respective  
6             addressing value of the first vector register,  
7             means for generating each respective data value of a first sequence of  
8             values by combining at least a portion of each respective addressing value  
9             of the first vector register to a respective one of a sequence of integer  
10            numbers,  
11            means for storing the first sequence of values to the constrained  
12            memory area using the generated sequence of respective address values,  
13            means for loading a second first sequence of values from the  
14            constrained memory area using the generated sequence of respective  
15            address values, and  
16            means for comparing the first sequence of values to the second  
17            sequence of values.

1     25.     The system of claim 21, further wherein the means for loading of the third

2 vector register of each processor includes means for loading elements from locations  
3 specified by addressing values corresponding to indications of positive compares  
4 from the comparing operation.

1 26. The system of claim 21, further wherein indirect addresses of the elements  
2 from memory are calculated by adding each respective addressing value to a base  
3 address.

1 27. A computer-readable medium having instructions stored thereon for causing  
2 a suitably programmed information-processing system to execute a method  
3 comprising:

4 loading a first vector register with addressing values;  
5 loading a second vector register with operand values;  
6 determining which, if any, element addresses of the first vector register have  
7 a value that duplicates a value in another element address;  
8 selectively adding certain elements of the second vector of operand values  
9 based on the element addresses the duplicated values;  
10 loading, using indirect addressing from the first vector register, elements  
11 from memory into a third vector register;  
12 adding values from the third vector register and the second vector of operand  
13 values to generate a result vector; and  
14 storing the result vector to memory using indirect addressing.